```
PPPPPPPPPPPP        AAAAAAAAA     TTTTTTTTTTTTTTT     CCCCCCCCCCC  HHH         HHH
PPPPPPPPPPPP        AAAAAAAAA     TTTTTTTTTTTTTTT     CCCCCCCCCCC  HHH         HHH
PPPPPPPPPPPP        AAAAAAAAA     TTTTTTTTTTTTTTT     CCCCCCCCCCC  HHH         HHH
PPP         PPP   AAA         AAA       TTT         CCC           HHH         HHH
PPP         PPP   AAA         AAA       TTT         CCC           HHH         HHH
PPP         PPP   AAA         AAA       TTT         CCC           HHH         HHH
PPP         PPP   AAA         AAA       TTT         CCC           HHH         HHH
PPP         PPP   AAA         AAA       TTT         CCC           HHH         HHH
PPP         PPP   AAA         AAA       TTT         CCC           HHH         HHH
PPPPPPPPPPPP      AAA         AAA       TTT         CCC           HHHHHHHHHHHHHHH
PPPPPPPPPPPP      AAA         AAA       TTT         CCC           HHHHHHHHHHHHHHH
PPPPPPPPPPPP      AAA         AAA       TTT         CCC           HHHHHHHHHHHHHHH
PPP               AAAAAAAAAAAAAAA       TTT         CCC           HHH         HHH
PPP               AAAAAAAAAAAAAAA       TTT         CCC           HHH         HHH
PPP               AAAAAAAAAAAAAAA       TTT         CCC           HHH         HHH
PPP               AAA         AAA       TTT         CCC           HHH         HHH
PPP               AAA         AAA       TTT         CCC           HHH         HHH
PPP               AAA         AAA       TTT         CCC           HHH         HHH
PPP               AAA         AAA       TTT         CCC           HHH         HHH
PPP               AAA         AAA       TTT            CCCCCCCCCCC  HHH         HHH
PPP               AAA         AAA       TTT            CCCCCCCCCCC  HHH         HHH
PPP               AAA         AAA       TTT            CCCCCCCCCCC  HHH         HHH
```

FILEID**PATARI

```
PPPPPPPP      AAAAAA    TTTTTTTTTT    AAAAAA    RRRRRRRR      IIIIII
PPPPPPPP      AAAAAA    TTTTTTTTTT    AAAAAA    RRRRRRRR      IIIIII
PP      PP   AA    AA       TT       AA    AA  RR      RR      II
PP      PP   AA    AA       TT       AA    AA  RR      RR      II
PP      PP   AA    AA       TT       AA    AA  RR      RR      II
PP      PP   AA    AA       TT       AA    AA  RR      RR      II
PPPPPPPP     AA    AA       TT       AA    AA  RRRRRRRR      II
PPPPPPPP     AA    AA       TT       AA    AA  RRRRRRRR      II
PP           AAAAAAAAAA     TT       AAAAAAAAAA  RR  RR      II
PP           AAAAAAAAAA     TT       AAAAAAAAAA  RR  RR      II
PP           AA    AA       TT       AA    AA  RR    RR      II          ....
PP           AA    AA       TT       AA    AA  RR    RR      II          ....
PP           AA    AA       TT       AA    AA  RR      RR    IIIIII      ....
PP           AA    AA       TT       AA    AA  RR      RR    IIIIII      ....

LL             IIIIII      SSSSSSSS
LL             IIIIII      SSSSSSSS
LL               II      SS
LL               II      SS
LL               II      SS
LL               II        SSSSSS
LL               II        SSSSSS
LL               II              SS
LL               II              SS
LL               II              SS
LL               II              SS
LLLLLLLLLL     IIIIII    SSSSSSSS
LLLLLLLLLL     IIIIII    SSSSSSSS
```

```
;     1          0001  0 MODULE PATARI (
;     2        L 0002  0                 %IF %VARIANT EQL 1
;     3          0003  0                 %THEN
;     4          0004  0                         ADDRESSING_MODE (EXTERNAL = LONG_RELATIVE, NONEXTERNAL = LONG_RELATIVE),
;     5          0005  0                 %FI
;     6          0006  0                 IDENT = 'V04-000') =
;     7          0007  1 BEGIN
;     8          0008  1
;     9          0009  1 !
;    10          0010  1 !*****************************************************************************
;    11          0011  1 !*                                                                           *
;    12          0012  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
;    13          0013  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
;    14          0014  1 !*  ALL RIGHTS RESERVED.                                                      *
;    15          0015  1 !*                                                                           *
;    16          0016  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
;    17          0017  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
;    18          0018  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
;    19          0019  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
;    20          0020  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
;    21          0021  1 !*  TRANSFERRED.                                                              *
;    22          0022  1 !*                                                                           *
;    23          0023  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
;    24          0024  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT     *
;    25          0025  1 !*  CORPORATION.                                                              *
;    26          0026  1 !*                                                                           *
;    27          0027  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
;    28          0028  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                   *
;    29          0029  1 !*                                                                           *
;    30          0030  1 !*                                                                           *
;    31          0031  1 !*****************************************************************************
;    32          0032  1 !
;    33          0033  1
;    34          0034  1 !++
;    35          0035  1 ! FACILITY:     PATCH
;    36          0036  1 !
;    37          0037  1 ! ABSTRACT:
;    38          0038  1 !
;    39          0039  1 !       processes names, displays expressions, and writes into memory.
;    40          0040  1 !
;    41          0041  1 ! ENVIRONMENT:  STARLET, user mode, interrupts disabled. non-AST level.
;    42          0042  1 !
;    43          0043  1 ! Version:      V02-014
;    44          0044  1 !
;    45          0045  1 ! History:
;    46          0046  1 !       Author:
;    47          0047  1 !               Carol Peters, 26 Oct 1976: Version 00
;    48          0048  1 !
;    49          0049  1 ! MODIFIED BY:
;    50          0050  1 !
;    51          0051  1 !       V03-002 MTR0015         Mike Rhodes             01-Nov-1982
;    52          0052  1 !               Modify routine PAT$WRITE_MEM to change the attributes
;    53          0053  1 !               of a patched DZRO image section to to be CRF.
;    54          0054  1 !
;    55          0055  1 !       V03-001 MTR0012         Mike Rhodes             16-Aug-1982
;    56          0056  1 !               Modify file names to remove duplicate file name useage
;    57          0057  1 !               between code and require files.
```

```
;    58          0058  1 !
;    59          0059  1 !       V02-014 KDM0043         Kathleen D. Morse       03-MAR-1981
;    60          0060  1 !               Dont create un-implemented format of ISD.
;    61          0061  1 !
;    62          0062  1 !       V02-013 PCG0001         Peter George            02-FEB-1981
;    63          0063  1 !               Add require statement for LIB$:PATDEF.REQ
;    64          0064  1 !
;    65          0065  1 !--
```

```
;    67          0066  1 FORWARD ROUTINE
;    68          0067  1           PAT$TRANS_NAME : NOVALUE,                  ! Transforms a string into a value
;    69          0068  1           PAT$MAP_ADDR : NOVALUE,                    ! Computes mapped address and maps image sec
;    70          0069  1           PAT$UNMAP_ADDR : NOVALUE,                  ! Computes unmapped address
;    71          0070  1           PAT$GET_VALUE : NOVALUE,                   ! Gets a stream of bytes from the image
;    72          0071  1           PAT$WRITE_MEM;                             ! Writes data into memory
;    73          0072  1
;    74          0073  1 LIBRARY 'SYS$LIBRARY:LIB.L32';
;    75          0074  1 REQUIRE 'SRC$:PATRTS.REQ';
;    76          1170  1 REQUIRE 'SRC$:PATPCT.REQ';
;    77          1210  1 REQUIRE 'SRC$:PATGEN.REQ';
;    78          1432  1 REQUIRE 'SRC$:BSTRUC.REQ';
;    79          1508  1 REQUIRE 'SRC$:DLLNAM.REQ';
;    80          1566  1 REQUIRE 'SRC$:VXSMAC.REQ';
;    81          1631  1 REQUIRE 'SRC$:PATTER.REQ';
;    82          1838  1 REQUIRE 'LIB$:PATDEF.REQ';                           ! Defines literals
;    83          1892  1 REQUIRE 'LIB$:PATMSG.REQ';
;    84          2066  1 REQUIRE 'SRC$:SYSSER.REQ';
```

```
;  R2098  1     SWITCHES LIST (SOURCE);
;  R2099  1
;  R210C  1     EXTERNAL ROUTINE
;  R2101  1              PAT$fao_out;            ! formats a line and outputs to the terminal
;  R2102  1
```

```
;    85          2148  1 REQUIRE 'SRC$:PREFIX.REQ';
;    86          2336  1 REQUIRE 'SRC$:PATPRE.REQ';
;    87          2499  1
;    88          2500  1 EXTERNAL ROUTINE
;    89          2501  1          PAT$CREMAP,                                    ! Create and maps image sections
;    90          2502  1          PAT$FIND_SYM;                                  ! Matches a name with a symbol
;    91          2503  1
;    92          2504  1 EXTERNAL
;    93          2505  1          PAT$GL_NEWVBNMX,                               ! Max VBN in new image used for image sectio
;    94          2506  1          PAT$GL_IMGBLKS,                                ! Number of blocks in new image
;    95          2507  1          PAT$GL_ISELHD,                                 ! Listhead for image section table
;    96          2508  1          PAT$GB_MOD_PTR: REF VECTOR [, BYTE],           ! Pointer to current modes
;    97          2509  1          PAT$GB_LOC_TYPE: BYTE,                         ! Type of last location examined
;    98          2510  1          PAT$GL_LAST_LOC,                               ! Current location
;    99          2511  1          PAT$GL_LAST_VAL,                               ! Current value
;   100          2512  1          PAT$GL_SEMAN1: VECTOR,                         ! First semantic stack, holds tokens
;   101          2513  1          PAT$GL_SEMAN2: VECTOR;                         ! Second semantic stack, holds string pointe
```

```
;  103     2514  1 GLOBAL ROUTINE PAT$TRANS_NAME (SEMSP, LEXEME_STG_DESC) : NOVALUE =
;  104     2515  1
;  105     2516  1 !++
;  106     2517  1 ! Functional description:
;  107     2518  1 !
;  108     2519  1 !       Transforms the simplest element of a PATCH expression
;  109     2520  1 !       into a binary value. Tokens expected are ALPHA_STR_TOKEN,
;  110     2521  1 !       DIGIT_STR_TOKEN, and the tokens for current location,
;  111     2522  1 !       last value displayed, next location, and previous location.
;  112     2523  1 !
;  113     2524  1 !       A name token is represented as a length count and a buffer
;  114     2525  1 !       address in the string descriptor. A number token is represented as
;  115     2526  1 !       a length count and a 32-bit or 64-bit precision number in
;  116     2527  1 !       the buffer address.
;  117     2528  1 !
;  118     2529  1 ! Calling sequence:
;  119     2530  1 !
;  120     2531  1 !       CALLS #2, PAT$TRANS_NAME
;  121     2532  1 !
;  122     2533  1 ! Inputs:
;  123     2534  1 !
;  124     2535  1 !       SEMSP           - offset in parse stack that holds the
;  125     2536  1 !                         current token.
;  126     2537  1 !       LEXEME_STG_DESC - string descriptor to number or name
;  127     2538  1 !
;  128     2539  1 ! Implicit inputs:
;  129     2540  1 !
;  130     2541  1 !       current mode, last value, current location, next location
;  131     2542  1 !
;  132     2543  1 ! Outputs:
;  133     2544  1 !
;  134     2545  1 !       none
;  135     2546  1 !
;  136     2547  1 ! Implicit outputs:
;  137     2548  1 !
;  138     2549  1 !       pushes a value onto the stack in the place of the token found
;  139     2550  1 !
;  140     2551  1 ! Routine value:
;  141     2552  1 !
;  142     2553  1 !       novalue
;  143     2554  1 !
;  144     2555  1 ! Side effects:
;  145     2556  1 !
;  146     2557  1 !       none
;  147     2558  1 !--
;  148     2559  1
;  149     2560  2 BEGIN
;  150     2561  2
;  151     2562  2 MAP
;  152     2563  2         LEXEME_STG_DESC : REF BLOCK [, BYTE];
;  153     2564  2
;  154     2565  2
;  155     2566  3 PAT$GL_SEMAN1 [.SEMSP] = (SELECTONE .PAT$GL_SEMAN1 [.SEMSP] OF
;  156     2567  3         SET
;  157     2568  3
;  158     2569  3         [DIGIT_STR_TOKEN]:
;  159     2570  3
```

```
;  160          2571  4                BEGIN
;  161          2572  5                .(.LEXEME_STG_DESC [DSC$A_POINTER])
;  162          2573  3                END;
;  163          2574
;  164          2575  3           [PERIOD_TOKEN]:
;  165          2576  4                BEGIN
;  166          2577  4                .PAT$GL_LAST_LOC
;  167          2578  3                END;
;  168          2579
;  169          2580  3           [UP_ARROW_TOKEN]:
;  170          2581  4                BEGIN
;  171          2582  4                .PAT$GL_LAST_LOC - .PAT$GB_MOD_PTR [MODE_LENGTH]
;  172          2583  3                END;
;  173          2584
;  174          2585  3           [BACKSLASH_TOKEN]:
;  175          2586  4                BEGIN
;  176          2587  4                .PAT$GL_LAST_VAL
;  177          2588  3                END;
;  178          2589
;  179          2590  3           [ALPHA_STR_TOKEN]:
;  180          2591  4                BEGIN
;  181          2592  4                LOCAL
;  182          2593  4                      INDEX;
;  183          2594  4                INDEX = PAT$FIND_SYM (.LEXEME_STG_DESC);
;  184          2595  5                IF (.INDEX NEQ 0)
;  185          2596  4                THEN
;  186          2597  5                      BEGIN
;  187          2598  5                      .SYM_VALUE (.INDEX)
;  188          2599  5                      END
;  189          2600  4                ELSE SIGNAL (PAT$_NOSUCHSYM)
;  190          2601  3                END;
;  191          2602
;  192          2603  3           [OTHERWISE]:
;  193          2604  3                SIGNAL (PAT$_PARSEERR);
;  194          2605  3
;  195          2606  2           TES);
;  196          2607  2
;  197          2608  1 END;


                                        .TITLE  PATARI
                                        .IDENT  \V04-000\

                                ISE$C_SIZE==            20
                                TXT$C_SIZE==            4
                                PAL$C_SIZE==            16
                                ASD$C_SIZE==            9
                                FWR$C_SIZE==            24
                                        .EXTRN  PAT$FAO_OUT, PAT$CREMAP
                                        .EXTRN  PAT$FIND_SYM, PAT$GL_NEWVBNMX
                                        .EXTRN  PAT$GL_IMGBLKS, PAT$GL_ISELHD
                                        .EXTRN  PAT$GB_MOD_PTR, PAT$GB_LOC_TYPE
                                        .EXTRN  PAT$GL_LAST_LOC
                                        .EXTRN  PAT$GL_LAST_VAL
                                        .EXTRN  PAT$GL_SEMAN1, PAT$GL_SEMAN2
                                        .WEAK   ACCESS_CHECK
```

```
                                                        .PSECT  _PAT$CODE,NOWRT,2

                                                001C 00000      .ENTRY  PAT$TRANS_NAME, Save R2,R3,R4                   ; 2514
                                    54 00000000G   EF   9E 00002    MOVAB   PAT$GL_SEMAN1, R4
                                    53 00000000G   EF   9E 00009    MOVAB   PAT$GL_LAST_LOC, R3
                                    52        04   AC   D0 00010    MOVL    SEMSP, R2                               ; 2566
                                    51      6442   D0   D0 00014    MOVL    PAT$GL_SEMAN1[R2], R1
                                00000048    8F        51   D1 00018    CMPL    R1, #72                                 ; 2570
                                                      0A   12 0001F    BNEQ    1$
                                    50        08   AC   D0 00021    MOVL    LEXEME_STG_DESC, R0                     ; 2572
                                    50        04   B0   D0 00025    MOVL    @4(R0), R0                              ; 2571
                                                      68   11 00029    BRB     8$
                                0000004B    8F        51   D1 0002B 1$:    CMPL    R1, #75                         ; 2575
                                                      05   12 00032    BNEQ    2$
                                    50              63   D0 00034    MOVL    PAT$GL_LAST_LOC, R0                     ; 2576
                                                      5A   11 00037    BRB     8$
                                00000053    8F        51   D1 00039 2$:    CMPL    R1, #83                         ; 2580
                                                      11   12 00040    BNEQ    3$
                                    50 00000000G   EF   D0 00042    MOVL    PAT$GB_MOD_PTR, R0                      ; 2582
                                    50        01   A0   9A 00049    MOVZBL  1(R0), R0
                            50                    63   C3 0004D    SUBL3   R0, PAT$GL_LAST_LOC, R0
                                                      40   11 00051    BRB     8$                                      ; 2581
                                    3E              51   D1 00053 3$:    CMPL    R1, #62                         ; 2585
                                                      09   12 00056    BNEQ    4$
                                    50 00000000G   EF   D0 00058    MOVL    PAT$GL_LAST_VAL, R0                     ; 2586
                                                      32   11 0005F    BRB     8$
                                00000047    8F        51   D1 00061 4$:    CMPL    R1, #71                         ; 2590
                                                      1C   12 00068    BNEQ    6$
                                                08    AC   DD 0006A    PUSHL   LEXEME_STG_DESC                         ; 2594
                                00000000G   EF        01   FB 0006D    CALLS   #1, PAT$FIND_SYM
                                                      50   D5 00074    TSTL    INDEX                                   ; 2595
                                                      06   13 00076    BEQL    5$
                                    50        08   A0   D0 00078    MOVL    8(INDEX), R0                            ; 2597
                                                      15   11 0007C    BRB     8$
                                          006D8088    8F   DD 0007E 5$:    PUSHL   #7176328                        ; 2600
                                                      06   11 00084    BRB     7$
                                          006D81A2    8F   DD 00086 6$:    PUSHL   #7176514                        ; 2604
                                00000000G   00        01   FB 0008C 7$:    CALLS   #1, LIB$SIGNAL
                                      6442            50   D0 00093 8$:    MOVL    R0, PAT$GL_SEMAN1[R2]           ; 2566
                                                      04   00097          RET                                     ; 2608

; Routine Size:  152 bytes,    Routine Base:  _PAT$CODE + 0000
```

```
;   199          2609  1 GLOBAL ROUTINE PAT$MAP_ADDR (UNMAPPED_ADDR, MAPPED_ADDR_PTR, ISE_ADDR_PTR) : NOVALUE =
;   200          2610  1
;   201          2611  1 !++
;   202          2612  1 ! FUNCTIONAL DESCRIPTION:
;   203          2613  1 !
;   204          2614  1 !       Determines the mapped address given an unmapped address in the image.
;   205          2615  1 !       The input parameter is the unmapped address and the output parameters
;   206          2616  1 !       are the mapped address and image section entry address.
;   207          2617  1 !
;   208          2618  1 !       First, the image section table is searched to find out if the address
;   209          2619  1 !       to be mapped is in the image.  If not, then an error message is produced
;   210          2620  1 !       and the appropriate action is taken by the error routine (control
;   211          2621  1 !       returns for next command or to CLI).  Then the image section is mapped
;   212          2622  1 !       into memory if it is not already there.  This may also produce an
;   213          2623  1 !       error message similar to the above.  Then the mapped address is
;   214          2624  1 !       computed and returned.
;   215          2625  1 !
;   216          2626  1 !       Then the mapped address is computed and returned.
;   217          2627  1 !
;   218          2628  1 ! CALLING SEQUENCE:
;   219          2629  1 !
;   220          2630  1 !       PAT$MAP_ADDR ()
;   221          2631  1 !
;   222          2632  1 ! INPUTS:
;   223          2633  1 !
;   224          2634  1 !       UNMAPPED_ADDR   - The unmapped address within the image
;   225          2635  1 !       MAPPED_ADDR_PTR - Place to store the corresponding mapped address
;   226          2636  1 !       ISE_ADDR_PTR    - Place to store the corresponding image section entry address
;   227          2637  1 !
;   228          2638  1 ! IMPLICIT INPUTS:
;   229          2639  1 !
;   230          2640  1 !       The image section table must have been set up.
;   231          2641  1 !
;   232          2642  1 ! OUTPUTS:
;   233          2643  1 !
;   234          2644  1 !       MAPPED_ADDR_PTR - The corresponding mapped address
;   235          2645  1 !       ISE_ADDR_PTR    - The corresponding image section entry address
;   236          2646  1 !
;   237          2647  1 ! IMPLICIT OUTPUTS:
;   238          2648  1 !
;   239          2649  1 !       NONE
;   240          2650  1 !
;   241          2651  1 ! ROUTINE VALUE:
;   242          2652  1 !
;   243          2653  1 !       NONE
;   244          2654  1 !
;   245          2655  1 ! SIDE EFFECTS:
;   246          2656  1 !
;   247          2657  1 !       The image section is mapped into memory if it was not before.
;   248          2658  1 !
;   249          2659  1 !--
;   250          2660  1
;   251          2661  2 BEGIN
;   252          2662  2
;   253          2663  2 MAP
;   254          2664  2         ISE_ADDR_PTR : REF VECTOR,                    ! Address of corresponding image section tab
;   255          2665  2         MAPPED_ADDR_PTR : REF VECTOR;                 ! Mapped address returned
```

```
;   256          2666  2
;   257          2667  2
;   258          2668  2 LOCAL
;   259          2669  2         CURRENT_ISE: REF BLOCK[,BYTE];                          ! Current image section entry during search
;   260          2670  2
;   261          2671  2 !++
;   262          2672  2 ! Initialize for search through image section table to find the image
;   263          2673  2 ! section containing the unmapped virtual address.
;   264          2674  2 !--
;   265          2675  2 ISE_ADDR_PTR[0]=0;                                              ! Initialize to none
;   266          2676  2 CURRENT_ISE=.PAT$GL_ISELHD;                                     ! Set listhead of image section table
;   267          2677  2
;   268          2678  2 !++
;   269          2679  2 ! Search through the image section table to find the image section which
;   270          2680  2 ! contains the unmapped virtual address.  Stop when the table runs out or the
;   271          2681  2 ! iamge section is found.
;   272          2682  2 !--
;   273          2683  3 WHILE (.CURRENT_ISE NEQA 0)
;   274          2684  2 DO
;   275          2685  3         BEGIN
;   276          2686  3         IF (.UNMAPPED_ADDR GEQA .CURRENT_ISE[ISE$L_IMGVST]) AND
;   277          2687  4            (.UNMAPPED_ADDR LEQA .CURRENT_ISE[ISE$L_IMGVEND])
;   278          2688  3         THEN
;   279          2689  4                 BEGIN
;   280          2690  4                 ISE_ADDR_PTR[0]=.CURRENT_ISE;                  ! Found starting image section
;   281          2691  4                 EXITLOOP;
;   282          2692  3                 END;
;   283          2693  3         CURRENT_ISE=.CURRENT_ISE[ISE$L_NXTISE];                ! Set to next ISE in list
;   284          2694  2         END;
;   285          2695  2
;   286          2696  2 !++
;   287          2697  2 ! Check that the address was within the image section.
;   288          2698  2 !--
;   289          2699  3 IF (.ISE_ADDR_PTR[0] EQLA 0)
;   290          2700  2 THEN
;   291          2701  2         SIGNAL(PAT$_NSADDR,1,.UNMAPPED_ADDR);                  ! Starting address is not within image, repo
;   292          2702  2
;   293          2703  2 !++
;   294          2704  2 ! Check that the image section is mapped.  If not, map it.  If it cannot
;   295          2705  2 ! be mapped, an error message is produced and this command is aborted with
;   296          2706  2 ! a possible image exit.
;   297          2707  2 !--
;   298          2708  3 IF (.CURRENT_ISE[ISE$L_MAPVEND] EQL 0)                         ! Is image section not mapped?
;   299          2709  2 THEN
;   300          2710  2         PAT$CREMAP(.ISE_ADDR_PTR[0]);                          ! Yes, then map the image section
;   301          2711  2
;   302          2712  2 !++
;   303          2713  2 ! Now compute the corresponding mapped address.
;   304          2714  2 !--
;   305          2715  2 MAPPED_ADDR_PTR[0] = .CURRENT_ISE[ISE$L_MAPVST] + (.UNMAPPED_ADDR - .CURRENT_ISE[ISE$L_IMGVST]);
;   306          2716  2 RETURN
;   307          2717  1 END;
```

```
                                                0004 00000          .ENTRY  PAT$MAP_ADDR, Save R2                           ; 2609
                                             0C BC   D4 00002       CLRL    @ISE_ADDR_PTR                                   ; 2675
                                      52 00000000G   EF   D0 00005  MOVL    PAT$GL_ISELHD, CURRENT_ISE                      ; 2676
                                                     19   13 0000C  1$:     BEQL    3$                                      ; 2683
                                   04   A2              04   AC   D1 0000E  CMPL    UNMAPPED_ADDR, 4(CURRENT_ISE)           ; 2686
                                                        0D   1F 00013  BLSSU   2$
                                   08   A2              04   AC   D1 00015  CMPL    UNMAPPED_ADDR, 8(CURRENT_ISE)           ; 2687
                                                        06   1A 0001A  BGTRU   2$
                                   0C   BC                  52   D0 0001C  MOVL    CURRENT_ISE, @ISE_ADDR_PTR              ; 2690
                                                            05   11 00020  BRB     3$                                      ; 2689
                                        52                  62   D0 00022  2$:     MOVL    (CURRENT_ISE), CURRENT_ISE      ; 2693
                                                            E5   11 00025  BRB     1$                                      ; 2683
                                                 0C     BC   D5 00027  3$:     TSTL    @ISE_ADDR_PTR                           ; 2699
                                                        12   12 0002A  BNEQ    4$
                                                 04     AC   DD 0002C  PUSHL   UNMAPPED_ADDR                                  ; 2701
                                                        01   DD 0002F  PUSHL   #1
                                          006D812A      8F   DD 00031  PUSHL   #7176490
                                00000000G   00          03   FB 00037  CALLS   #3, LIB$SIGNAL
                                                 10     A2   D5 0003E  4$:     TSTL    16(CURRENT_ISE)                         ; 2708
                                                        0A   12 00041  BNEQ    5$                                      ; 2710
                                                 0C     BC   DD 00043  PUSHL   @ISE_ADDR_PTR
                                00000000G       EF          01   FB 00046  CALLS   #1, PAT$CREMAP                          ; 2715
                             50         04     AC       04   A2   C3 0004D  5$:     SUBL3   4(CURRENT_ISE), UNMAPPED_ADDR, R0
                                        08     BC       0C B240   9E 00053  MOVAB   @12(CURRENT_ISE)[R0], @MAPPED_ADDR_PTR
                                                                04 00059  RET                                             ; 2717

; Routine Size:  90 bytes,    Routine Base:  _PAT$CODE + 0098
```

```
;  309     2718  1 GLOBAL ROUTINE PAT$UNMAP_ADDR (UNMAP_ADDR_PTR, MAPPED_ADDR, ISE_ADDR_PTR) : NOVALUE =
;  310     2719  1
;  311     2720  1 !++
;  312     2721  1 ! FUNCTIONAL DESCRIPTION:
;  313     2722  1 !
;  314     2723  1 !       Determines the unmapped address given a mapped address in the image.
;  315     2724  1 !       The input parameter is the mapped address and the output parameters
;  316     2725  1 !       are the unmapped address and the image section table entry address.
;  317     2726  1 !
;  318     2727  1 !       First, the image section table is searched to find out if the address
;  319     2728  1 !       to be unmapped is in the image.  If not, then an error message is
;  320     2729  1 !       produced and the apporpriate action is taken by the error routine
;  321     2730  1 !       (control returns for next command or to CLI).
;  322     2731  1 !
;  323     2732  1 !       Then the unmapped address is computed and returned.
;  324     2733  1 !
;  325     2734  1 !       If everything was successful, the routine returns TRUE.
;  326     2735  1 !
;  327     2736  1 ! CALLING SEQUENCE:
;  328     2737  1 !
;  329     2738  1 !       PAT$UNMAP_ADDR ()
;  330     2739  1 !
;  331     2740  1 ! INPUTS:
;  332     2741  1 !
;  333     2742  1 !       UNMAP_ADDR_PTR  - Place to store the corresponding unmapped address
;  334     2743  1 !       MAPPED_ADDR     - The mapped address
;  335     2744  1 !       ISE_ADDR_PTR    - Place to store the corresponding image section entry address
;  336     2745  1 !
;  337     2746  1 ! IMPLICIT INPUTS:
;  338     2747  1 !
;  339     2748  1 !       The image section table must have been set up.
;  340     2749  1 !
;  341     2750  1 ! OUTPUTS:
;  342     2751  1 !
;  343     2752  1 !       UNMAP_ADDR_PTR  - The corresponding unmapped address
;  344     2753  1 !       ISE_ADDR_PTR    - The corresponding image section entry address
;  345     2754  1 !
;  346     2755  1 ! IMPLICIT OUTPUTS:
;  347     2756  1 !
;  348     2757  1 !       none
;  349     2758  1 !
;  350     2759  1 ! ROUTINE VALUE:
;  351     2760  1 !
;  352     2761  1 !       none
;  353     2762  1 !
;  354     2763  1 ! SIDE EFFECTS:
;  355     2764  1 !
;  356     2765  1 !       none
;  357     2766  1 !
;  358     2767  1 !--
;  359     2768  1
;  360     2769  2 BEGIN
;  361     2770  2
;  362     2771  2 MAP
;  363     2772  2         UNMAP_ADDR_PTR  : REF VECTOR,                   ! Unmapped address to find
;  364     2773  2         ISE_ADDR_PTR : REF VECTOR;                      ! Address of corresponding image section tab
;  365     2774  2
```

```
;  366          2775  2 LOCAL
;  367          2776  2         CURRENT_ISE: REF BLOCK[,BYTE];                          ! Current image section entry during search
;  368          2777  2
;  369          2778  2 !++
;  370          2779  2 ! Initialize for search through image section table to find the image
;  371          2780  2 ! section containing the mapped virtual address.
;  372          2781  2 !--
;  373          2782  2 ISE_ADDR_PTR[0]=0;                                              ! Initialize to none
;  374          2783  2 CURRENT_ISE=.PAT$GL_ISELHD;                                     ! Set listhead of image section table
;  375          2784  2
;  376          2785  2 !++
;  377          2786  2 ! Search through the image section table to find the image section which
;  378          2787  2 ! contains the mapped virtual address.  Stop when the table runs out or the
;  379          2788  2 ! image section is found.
;  380          2789  2 !--
;  381          2790  3 WHILE (.CURRENT_ISE NEQA 0)
;  382          2791  2 DO
;  383          2792  3         BEGIN
;  384          2793  3         IF (.MAPPED_ADDR GEQA .CURRENT_ISE[ISE$L_MAPVST]) AND
;  385          2794  4            (.MAPPED_ADDR LEQA .CURRENT_ISE[ISE$L_MAPVEND])
;  386          2795  3         THEN
;  387          2796  4                 BEGIN
;  388          2797  4                 ISE_ADDR_PTR[0]=.CURRENT_ISE;                  ! Found starting image section
;  389          2798  4                 EXITLOOP;
;  390          2799  3                 END;
;  391          2800  3         CURRENT_ISE=.CURRENT_ISE[ISE$L_NXTISE];                ! Set to next ISE in list
;  392          2801  2         END;
;  393          2802  2
;  394          2803  2 !++
;  395          2804  2 ! Check that the address was within the image section.
;  396          2805  2 !--
;  397          2806  3 IF (.ISE_ADDR_PTR[0] EQLA 0)
;  398          2807  2 THEN
;  399          2808  2         SIGNAL(PAT$_PATERR);                                   ! Starting address is not within image, repo
;  400          2809  2
;  401          2810  2 !++
;  402          2811  2 ! Now compute the corresponding unmapped address.
;  403          2812  2 !--
;  404          2813  2 UNMAP_ADDR_PTR[0] = .CURRENT_ISE[ISE$L_IMGVST] + (.MAPPED_ADDR - .CURRENT_ISE[ISE$L_MAPVST]);
;  405          2814  2 RETURN
;  406          2815  1 END;


                                0004 00000          .ENTRY  PAT$UNMAP_ADDR, Save R2                 ; 2718
                          0C    BC   D4 00002         CLRL    @ISE_ADDR_PTR                         ; 2782
                52 00000000G    EF   D0 00005         MOVL    PAT$GL_ISELHD, CURRENT_ISE            ; 2783
                                19   13 00009 1$:      BEQL    3$                                    ; 2790
          0C    A2        08    AC   D1 0000B         CMPL    MAPPED_ADDR, 12(CURRENT_ISE)          ; 2793
                                0D   1F 00010         BLSSU   2$
          10    A2        08    AC   D1 00012         CMPL    MAPPED_ADDR, 16(CURRENT_ISE)          ; 2794
                                06   1A 00017         BGTRU   2$
          0C    BC              52   D0 00019         MOVL    CURRENT_ISE, @ISE_ADDR_PTR            ; 2797
                                05   11 0001D         BRB     3$                                    ; 2796
                52              62   D0 0001F 2$:     MOVL    (CURRENT_ISE), CURRENT_ISE            ; 2800
```

```
                                        E5  11 00025                BRB     1$                                              ; 2790
                                     0C BC  D5 00027 3$:            TSTL    @ISE_ADDR_PTR                                   ; 2806
                                        0D  12 0002A                BNEQ    4$                                              ;
                                 006D814A   8F  DD 0002C            PUSHL   #7176522                                        ; 2808
                  00000000G      00         01  FB 00032            CALLS   #1, LIB$SIGNAL                                  ;
                50      08   AC        0C   A2  C3 00039 4$:        SUBL3   12(CURRENT_ISE), MAPPED_ADDR, R0                ; 2813
                        04   BC        04 B240  9E 0003F            MOVAB   @4(CURRENT_ISE)[R0], @UNMAP_ADDR_PTR            ;
                                            04  00045               RET                                                     ; 2815

; Routine Size:  70 bytes,    Routine Base:  _PAT$CODE + 00F2
```

```
;  408     2816  1 GLOBAL ROUTINE PAT$GET_VALUE (UNMAPPED_ADDR, NUM_OF_BYTES, RETURN_ADDR_PTR) : NOVALUE =
;  409     2817  1
;  410     2818  1 !++
;  411     2819  1 ! FUNCTIONAL DESCRIPTION:
;  412     2820  1 !
;  413     2821  1 !       This routine takes an unmapped address and a count of bytes and
;  414     2822  1 !       returns the values of the stream described.
;  415     2823  1 !
;  416     2824  1 !       First, the starting and ending addresses of the stream are mapped.
;  417     2825  1 !       Then the number of bytes within the starting image section are moved
;  418     2826  1 !       into the return storage area.  If the stream was entirely within one
;  419     2827  1 !       image section, the routine is finished and returns.  If the ending
;  420     2828  1 !       image section is different from the starting image section, then the
;  421     2829  1 !       unmapped address of the next byte to be found is mapped to produce a
;  422     2830  1 !       new starting image section and mapped address.  The process repeats
;  423     2831  1 !       starting with a computation of the number of bytes within this image
;  424     2832  1 !       section.
;  425     2833  1 !
;  426     2834  1 !       If the stream is not entirely within the image, then the appropriate
;  427     2835  1 !       error message is produced and this patch command is ended.
;  428     2836  1 !
;  429     2837  1 ! CALLING SEQUENCE:
;  430     2838  1 !
;  431     2839  1 !       PAT$GET_VALUE ()
;  432     2840  1 !
;  433     2841  1 ! INPUTS:
;  434     2842  1 !
;  435     2843  1 !       UNMAPPED_ADDR   - The unmapped address for the byte stream
;  436     2844  1 !       NUM_OF_BYTES    - The number of bytes to be found in the stream
;  437     2845  1 !       RETURN_ADDR_PTR - Pointer to return storage area
;  438     2846  1 !
;  439     2847  1 ! IMPLICIT INPUTS:
;  440     2848  1 !
;  441     2849  1 !       The image section table must have been set up.
;  442     2850  1 !
;  443     2851  1 ! OUTPUTS:
;  444     2852  1 !
;  445     2853  1 !       none
;  446     2854  1 !
;  447     2855  1 ! IMPLICIT OUTPUTS:
;  448     2856  1 !
;  449     2857  1 !       The returned storage area contains the desired values.
;  450     2858  1 !
;  451     2859  1 ! ROUTINE VALUE:
;  452     2860  1 !
;  453     2861  1 !       none
;  454     2862  1 !
;  455     2863  1 ! SIDE EFFECTS:
;  456     2864  1 !
;  457     2865  1 !       The image section is mapped if it was not before.
;  458     2866  1 !
;  459     2867  1 !--
;  460     2868  1
;  461     2869  2 BEGIN
;  462     2870  2
;  463     2871  2 MAP
;  464     2872  2         RETURN_ADDR_PTR : REF VECTOR[,BYTE];                    ! Address of return storage area for byte st
```

```
;  465     2873  2
;  466     2874  2 LOCAL
;  467     2875  2         LENGTH,                                         ! Remaining length of byte stream to move
;  468     2876  2         PARTIAL_LENGTH,                                 ! Number of bytes within starting image sect
;  469     2877  2         VALUE_BUFFER,                                   ! Holds address of return value buffer
;  470     2878  2         ST_MAPPED_ADDR: REF VECTOR[,BYTE],              ! Starting mapped address of remaining byte
;  471     2879  2         END_MAPPED_ADDR,                                ! Ending mapped address of byte stream
;  472     2880  2         START_ISE: REF BLOCK[,BYTE],                    ! Starting image section for remaining byte
;  473     2881  2         END_ISE: REF BLOCK[,BYTE];                      ! Ending image section for remaining byte st
;  474     2882  2
;  475     2883  2 !++
;  476     2884  2 ! Initialize number of bytes left to move.  Find the starting and ending
;  477     2885  2 ! mapped addresses and image sections.
;  478     2886  2 !--
;  479     2887  2 LENGTH=.NUM_OF_BYTES;
;  480     2888  2 VALUE_BUFFER=.RETURN_ADDR_PTR;
;  481     2889  2 PAT$MAP_ADDR(.UNMAPPED_ADDR, ST_MAPPED_ADDR, START_ISE);
;  482     2890  2 PAT$MAP_ADDR(.UNMAPPED_ADDR+.NUM_OF_BYTES-1, END_MAPPED_ADDR, END_ISE);
;  483     2891  2
;  484     2892  2 !++
;  485     2893  2 ! This loop moves the bytes into the return storage area.  It only takes
;  486     2894  2 ! values from the starting image section.  If the byte stream is in more than
;  487     2895  2 ! one image section, a partial length is moved in, a new starting image section
;  488     2896  2 ! is found, and then the next partial length is moved into the return buffer.
;  489     2897  2 ! This is repeated until all the byte stream requested is moved.
;  490     2898  2 !--
;  491     2899  2 REPEAT
;  492     2900  3         BEGIN
;  493     2901  3
;  494     2902  3         !++
;  495     2903  3         ! Find the number of bytes within the starting image section.
;  496     2904  3         !--
;  497     2905  4         IF (.START_ISE EQLA .END_ISE)
;  498     2906  3         THEN
;  499     2907  3                 PARTIAL_LENGTH=.LENGTH
;  500     2908  3         ELSE
;  501     2909  3                 PARTIAL_LENGTH=.START_ISE[ISE$L_MAPVEND] - .ST_MAPPED_ADDR
;  502     2910  3                                 - .NUM_OF_BYTES + .LENGTH +1;
;  503     2911  3
;  504     2912  3         !++
;  505     2913  3         ! Move in the partial byte stream found in the starting image section.
;  506     2914  3         ! Also update the remaining length to be moved.
;  507     2915  3         !--
;  508     2916  3         LENGTH=.LENGTH - .PARTIAL_LENGTH;
;  509     2917  3         VALUE_BUFFER=CH$MOVE(.PARTIAL_LENGTH, .ST_MAPPED_ADDR, .VALUE_BUFFER);
;  510     2918  3
;  511     2919  3         !++
;  512     2920  3         ! Now check if all of the desired stream has been found.  If not, find
;  513     2921  3         ! a new starting image section and repeat the process.
;  514     2922  3         !--
;  515     2923  4         IF (.LENGTH EQL 0)
;  516     2924  3         THEN
;  517     2925  3                 RETURN;
;  518     2926  3         PAT$MAP_ADDR(.UNMAPPED_ADDR+.NUM_OF_BYTES-.LENGTH, ST_MAPPED_ADDR, START_ISE);
;  519     2927  3
;  520     2928  3
;  521     2929  2 END;                                                    ! End of loop to move byte stream
```

```
;  522          2930  1 END;                                                  ! End of PAT$GET_VALUE


                                    03FC 00000          .ENTRY  PAT$GET_VALUE, Save R2,R3,R4,R5,R6,R7,R8,R9 ; 2816
                              59 FF5A   CF  9E 00002    MOVAB   PAT$MAP_ADDR, R9                            ;
                              5E        10  C2 00007    SUBL2   #16, SP                                     ;
                              56    08  AC  D0 0000A    MOVL    NUM_OF_BYTES, LENGTH                        ; 2887
                              53    0C  AC  D0 0000E    MOVL    RETURN_ADDR_PTR, VALUE_BUFFER               ; 2888
                                    08  AE  9F 00012    PUSHAB  START_ISE                                   ; 2889
                                    10  AE  9F 00015    PUSHAB  ST_MAPPED_ADDR                              ;
                                    04  AC  DD 00018    PUSHL   UNMAPPED_ADDR                               ;
                              69        03  FB 0001B    CALLS   #3, PAT$MAP_ADDR                            ;
                                        5E  DD 0001E    PUSHL   SP                                          ; 2890
                                    08  AE  9F 00020    PUSHAB  END_MAPPED_ADDR                             ;
                        57    04  AC      08  AC  C1 00023    ADDL3   NUM_OF_BYTES, UNMAPPED_ADDR, R7         ;
                                    FF  A7  9F 00029    PUSHAB  -1(R7)                                      ;
                              69        03  FB 0002C 1$:   CALLS   #3, PAT$MAP_ADDR                        ;
                              50    08  AE  D0 0002F    MOVL    START_ISE, R0                               ; 2905
                              6E        50  D1 00033    CMPL    R0, END_ISE                                 ;
                                        05  12 00036    BNEQ    2$                                          ;
                              58        56  D0 00038    MOVL    LENGTH, PARTIAL_LENGTH                      ; 2907
                                        0F  11 0003B    BRB     3$                                          ;
                        50    10  A0      0C  AE  C3 0003D 2$:   SUBL3   ST_MAPPED_ADDR, 16(R0), R0         ; 2909
                              50    08  AC  C2 00043    SUBL2   NUM_OF_BYTES, R0                            ; 2910
                              58    01 A640   9E 00047    MOVAB   1(LENGTH)[R0], PARTIAL_LENGTH               ;
                              56        58  C2 0004C 3$:   SUBL2   PARTIAL_LENGTH, LENGTH                   ; 2916
                        63    0C  BE      58  28 0004F    MOVC3   PARTIAL_LENGTH, @ST_MAPPED_ADDR, -          ; 2917
                                                                  (VALUE_BUFFER)
                                        56  D5 00054    TSTL    LENGTH                                      ; 2923
                                        0C  13 00056    BEQL    4$                                          ;
                                    08  AE  9F 00058    PUSHAB  START_ISE                                   ; 2926
                                    10  AE  9F 0005B    PUSHAB  ST_MAPPED_ADDR                              ;
                        7E              57  56  C3 0005E    SUBL3   LENGTH, R7, -(SP)                       ;
                                        C8  11 00062    BRB     1$                                          ;
                                            04 00064 4$:    RET                                             ; 2930

; Routine Size:  101 bytes,    Routine Base:  _PAT$CODE + 0138
```

```
;  524     2931  1 GLOBAL ROUTINE PAT$WRITE_MEM (DEST_UNMAP_ADDR, SRC_ADDRESS, LENGTH) =
;  525     2932  1
;  526     2933  1 !++
;  527     2934  1 ! FUNCTIONAL DESCRIPTION:
;  528     2935  1 !
;  529     2936  1 !       Writes a sequence of values (bytes) to memory in
;  530     2937  1 !       the user program.  The mapped destination address, image section
;  531     2938  1 !       entry, source, and number of bytes to write are all passed as parameters.
;  532     2939  1 !
;  533     2940  1 !
;  534     2941  1 !       First, the image section table is searched to find out if the
;  535     2942  1 !       addresses to be modified are in the image.  If not, then an error
;  536     2943  1 !       message is produced and the appropriate action is taken by the
;  537     2944  1 !       error routine (control returns for next command or to CLI).  Then,
;  538     2945  1 !       the image section is mapped into memory if it is not already there.
;  539     2946  1 !       This may also produce an error message similar to the above.
;  540     2947  1 !
;  541     2948  1 !       Once the starting address is mapped, the routine writes out as much
;  542     2949  1 !       of the sequence contained in that image section.  If some of the
;  543     2950  1 !       sequence is in another section, the routine sets up new parameters and
;  544     2951  1 !       calls itself recursively to write the remaining bytes.
;  545     2952  1 !
;  546     2953  1 !       If everything was successful, the routine returns TRUE.
;  547     2954  1 !
;  548     2955  1 ! CALLING SEQUENCE:
;  549     2956  1 !
;  550     2957  1 !       PAT$WRITE_MEM ()
;  551     2958  1 !
;  552     2959  1 ! INPUTS:
;  553     2960  1 !
;  554     2961  1 !       DEST_UNMAP_ADDR - The address of the location to be changed
;  555     2962  1 !       START_ISE       - The address of the image section descriptor for
;  556     2963  1 !                         the first address to be written
;  557     2964  1 !       SRC_ADDRESS     - The address of where the bytes are stored.
;  558     2965  1 !       LENGTH          - The number of bytes to be written.
;  559     2966  1 !
;  560     2967  1 ! IMPLICIT INPUTS:
;  561     2968  1 !
;  562     2969  1 !       The image section table must be set up.
;  563     2970  1 !
;  564     2971  1 ! OUTPUTS:
;  565     2972  1 !
;  566     2973  1 !       TRUE or an error message and unwind/exit.
;  567     2974  1 !
;  568     2975  1 ! IMPLICIT OUTPUTS:
;  569     2976  1 !
;  570     2977  1 !       none
;  571     2978  1 !
;  572     2979  1 ! ROUTINE VALUE:
;  573     2980  1 !
;  574     2981  1 !       TRUE
;  575     2982  1 !
;  576     2983  1 ! SIDE EFFECTS:
;  577     2984  1 !
;  578     2985  1 !       The value is written to memory.
;  579     2986  1 !
;  580     2987  1 !       If the image section that was patched is a demand zero image section,
```

```
;  581          2988  1 !       then its attributes are converted to be a copy on reference section.
;  582          2989  1 !--
;  583          2990  1
;  584          2991  2 BEGIN
;  585          2992  2
;  586          2993  2 MAP
;  587          2994  2         DEST_UNMAP_ADDR : REF VECTOR[,BYTE],
;  588          2995  2         SRC_ADDRESS     : REF VECTOR[,BYTE];
;  589          2996  2
;  590          2997  2 LOCAL
;  591          2998  2         START_ISE: REF BLOCK[,BYTE],                        ! Image section table entry for starting add
;  592          2999  2         START_ISD: REF BLOCK[,BYTE],                        ! Image section descriptor for starting addr
;  593          3000  2         END_ISE,                                            ! Image section table entry for ending addre
;  594          3001  2         END_MAPPED_ADDR,                                    ! Mapped ending address to be altered
;  595          3002  2         END_UNMAP_ADDR,                                     ! Unmapped ending address to be altered
;  596          3003  2         DEST_MAPPED_ADR: REF VECTOR[,BYTE],                 ! Unmapped starting address to be altered
;  597          3004  2         PARTIAL_LENGTH;                                     ! Length to write into starting image sectio
;  598          3005  2
;  599          3006  2 !++
;  600          3007  2 ! Initialize for search through image section table to find the image
;  601          3008  2 ! sections containing the starting and ending virtual addresses to be altered.
;  602          3009  2 !--
;  603          3010  2 PAT$MAP_ADDR(.DEST_UNMAP_ADDR, DEST_MAPPED_ADR, START_ISE);   ! Find unmapped starting address
;  604          3011  2 END_UNMAP_ADDR=.DEST_UNMAP_ADDR + .LENGTH - 1;                ! Get last address to be written
;  605          3012  2
;  606          3013  2 !++
;  607          3014  2 ! Find the mapped ending address to be altered.  This will map the image
;  608          3015  2 ! section if it is not already mapped.
;  609          3016  2 !--
;  610          3017  2 PAT$MAP_ADDR(.END_UNMAP_ADDR, END_MAPPED_ADDR, END_ISE);     ! Map the ending address
;  611          3018  2
;  612          3019  2 !++
;  613          3020  2 ! Check that both addresses were within image sections.
;  614          3021  2 !--
;  615          3022  3 IF (.START_ISE EQLA 0)
;  616          3023  2 THEN
;  617          3024  2         SIGNAL(PAT$_NSADDR,1,.DEST_UNMAP_ADDR);             ! Starting address is not within image, repo
;  618          3025  3 IF (.END_ISE EQLA 0)
;  619          3026  2 THEN
;  620          3027  2         SIGNAL(PAT$_NSADDR,1,.END_UNMAP_ADDR);              ! Ending address is not within image, report
;  621          3028  2
;  622          3029  2 !++
;  623          3030  2 ! Now check if all of addresses to be altered are within the same image
;  624          3031  2 ! section.  If not, then set the length to be altered in this image section.
;  625          3032  2 !--
;  626          3033  3 IF (.START_ISE NEQA .END_ISE)
;  627          3034  2 THEN
;  628          3035  2         PARTIAL_LENGTH=.START_ISE[ISE$L_MAPVEND] - .DEST_MAPPED_ADR + 1
;  629          3036  2 ELSE
;  630          3037  2         PARTIAL_LENGTH=.LENGTH;
;  631          3038  2
;  632          3039  2 !++
;  633          3040  2 ! Move the new values into this image section.
;  634          3041  2 !--
;  635          3042  2 CH$MOVE(.PARTIAL_LENGTH,SRC_ADDRESS[0],DEST_MAPPED_ADR[0]);
;  636          3043  2
;  637          3044  2 !++
```

```
;  638          3045  2 ! Check if the image section was demand zero pages.  If so, change the image
;  639          3046  2 ! section descriptor to be a process private type.  The virtual block number is
;  640          3047  2 ! set to zero and must be changed when the new image is written out.  The image
;  641          3048  2 ! identification is set to zero.  The image section descriptor size is
;  642          3049  2 ! incremented to include the additional VBN field.
;  643          3050  2 !--
;  644          3051  2 START_ISD=.START_ISE + ISE$C_SIZE;                          ! Get address of image section descriptor
;  645          3052  2 IF .START_ISD[ISD$V_DZRO]                                   ! Check if was dmzro
;  646          3053  2 THEN
;  647          3054  2         BEGIN
;  648          3055  2         START_ISD[ISD$V_DZRO]=FALSE;                       ! Set no longer demand zero
;  649          3056  2         START_ISD[ISD$V_CRF] =TRUE;                        ! Set copy on reference
;  650          3057  2         START_ISD[ISD$L_VBN]=.PAT$GL_NEWVBNMX + 1;         ! Record unknown VBN
;  651          3058  2         START_ISD[ISD$W_SIZE]=.START_ISD[ISD$W_SIZE] + A_LONGWORD; ! Increment image section descriptor size
;  652          3059  2 !++
;  653          3060  2 ! These are changes for when process-private ISD's contain IDENT fields.
;  654          3061  2 ! This is currently an un-implemented format of ISD.
;  655          3062  2 !--
;  656          3063  2 !       START_ISD[ISD$L_IDENT]=0;                          ! Set ident to zero
;  657          3064  2 !       START_ISD[ISD$W_SIZE]=.START_ISD[ISD$W_SIZE] + A_QUADWORD; ! Increment image section descriptor size
;  658          3065  2         PAT$GL_NEWVBNMX = .PAT$GL_NEWVBNMX + .START_ISD[ISD$W_PAGCNT]; ! Increment max VBN in new image file
;  659          3066  2         PAT$GL_IMGBLKS = .PAT$GL_IMGBLKS + .START_ISD[ISD$W_PAGCNT]; ! Increment for number of new blocks in
;  660          3067  2         END;
;  661          3068  2
;  662          3069  2 !++
;  663          3070  2 ! Check if this was a global section.  If so, warn that only the local
;  664          3071  2 ! version is being patched.
;  665          3072  2 !--
;  666          3073  2 IF .START_ISD[ISD$V_GBL]
;  667          3074  2 THEN
;  668          3075  2         SIGNAL((PAT$_GBLWARN AND NOT STS$M_SEVERITY)+MSG$K_INFO,
;  669          3076  2                         1,START_ISD[ISD$T_GBLNAM]);        ! Only warn of possible problem
;  670          3077  2
;  671          3078  2 !++
;  672          3079  2 ! Check if all the values were changed.  If not, then recursively call
;  673          3080  2 ! this routine with new parameters.
;  674          3081  2 !--
;  675          3082  2 IF (.LENGTH EQL .PARTIAL_LENGTH)
;  676          3083  2 THEN
;  677          3084  2         RETURN TRUE
;  678          3085  2 ELSE
;  679          3086  2         BEGIN
;  680          3087  2         PAT$WRITE_MEM(DEST_UNMAP_ADDR[.PARTIAL_LENGTH],
;  681          3088  2                       SRC_ADDRESS[.PARTIAL_LENGTH],
;  682          3089  2                       .LENGTH-.PARTIAL_LENGTH);
;  683          3090  2         RETURN TRUE;
;  684          3091  2         END;
;  685          3092  2
;  686          3093  1 END;


                                    07FC 00000          .ENTRY  PAT$WRITE_MEM, Save R2,R3,R4,R5,R6,R7,R8,-  ; 2931
                                                                R9,R10                                          ;
                5A 00000000G   EF   9E 00002          MOVAB   PAT$GL_NEWVBNMX, R10                            ;
```

```
                        59     FEEE   CF  9E 00009          MOVAB   PAT$MAP_ADDR, R9                                 ;
                        58 00000000G   00  9E 0000E          MOVAB   LIB$SIGNAL, R8                                  ;
                        5E             10  C2 00015          SUBL2   #16, SP                                         ;
                                       5E  DD 00018          PUSHL   SP                                              ; 3010
                                 08    AE  9F 0001A          PUSHAB  DEST_MAPPED_ADR                                 ;
                                 04    AC  DD 0001D          PUSHL   DEST_UNMAP_ADDR                                 ;
                        69             03  FB 00020          CALLS   #3, PAT$MAP_ADDR                                ;
                52      04     AC       0C  AC  C1 00023     ADDL3   LENGTH, DEST_UNMAP_ADDR, R2                     ; 3011
                                 08    AE  9F 00029          PUSHAB  END_ISE                                         ; 3017
                                 10    AE  9F 0002C          PUSHAB  END_MAPPED_ADDR                                 ;
                                       72  9F 0002F          PUSHAB  -(END_UNMAP_ADDR)                               ;
                        69             03  FB 00031          CALLS   #3, PAT$MAP_ADDR                                ;
                        57             6E  D0 00034          MOVL    START_ISE, R7                                   ; 3022
                                       0E  12 00037          BNEQ    1$                                              ;
                                 04    AC  DD 00039          PUSHL   DEST_UNMAP_ADDR                                 ; 3024
                                       01  DD 0003C          PUSHL   #1                                              ;
                           006D812A    8F  DD 0003E          PUSHL   #7176490                                        ;
                        68             03  FB 00044          CALLS   #3, LIB$SIGNAL                                  ;
                                 08    AE  D5 00047 1$:      TSTL    END_ISE                                         ; 3025
                                       0D  12 0004A          BNEQ    2$                                              ;
                                       52  DD 0004C          PUSHL   END_UNMAP_ADDR                                  ; 3027
                                       01  DD 0004E          PUSHL   #1                                              ;
                           006D812A    8F  DD 00050          PUSHL   #7176490                                        ;
                        68             03  FB 00056          CALLS   #3, LIB$SIGNAL                                  ;
                08      AE             57  D1 00059 2$:      CMPL    R7, END_ISE                                     ; 3033
                                       0C  13 0005D          BEQL    3$                                              ;
                50      10     A7       04  AE  C3 0005F     SUBL3   DEST_MAPPED_ADR, 16(R7), R0                     ; 3035
                        56             01  A0  9E 00065     MOVAB   1(R0), PARTIAL_LENGTH                           ;
                                       04  11  00069         BRB     4$                                              ;
                        56       0C    AC  D0 0006B 3$:      MOVL    LENGTH, PARTIAL_LENGTH                          ; 3037
          04    BE       08     BC          56  28 0006F 4$:  MOVC3   PARTIAL_LENGTH, @SRC_ADDRESS, -               ; 3042
                                                               @DEST_MAPPED_ADR                                        ;
                        50       14    A7  9E 00075          MOVAB   20(R7), START_ISD                               ; 3051
                22      08     A0          02  E1 00079      BBC     #2, 8(START_ISD), 5$                            ; 3052
                        08     A0          04  8A 0007E      BICB2   #4, 8(START_ISD)                                ; 3055
                        08     A0          02  88 00082      BISB2   #2, 8(START_ISD)                                ; 3056
          0C    A0               6A        01  C1 00086      ADDL3   #1, PAT$GL_NEWVBNMX, 12(START_ISD)              ; 3057
                                 60        04  A0 0008B      ADDW2   #4, (START_ISD)                                 ; 3058
                                 51  02    A0  3C 0008E      MOVZWL  2(START_ISD), R1                                ; 3065
                                 6A        51  C0 00092      ADDL2   R1, PAT$GL_NEWVBNMX                             ;
                                 51  02    A0  3C 00095      MOVZWL  2(START_ISD), R1                                ; 3066
                   00000000G     EF        51  C0 00099      ADDL2   R1, PAT$GL_IMGBLKS                              ;
                                 0E    08  A0  E9 000A0 5$:  BLBC    8(START_ISD), 6$                                ; 3073
                                       14  A0  9F 000A4      PUSHAB  20(START_ISD)                                   ; 3076
                                       01  DD 000A7          PUSHL   #1                                              ;
                              006D8073     8F  DD 000A9      PUSHL   #7176307                                        ;
                                 68        03  FB 000AF      CALLS   #3, LIB$SIGNAL                                  ;
                                 56  0C    AC  D1 000B2 6$:  CMPL    LENGTH, PARTIAL_LENGTH                          ; 3082
                                       12  13  000B6         BEQL    7$                                              ;
                7E       0C    AC          56  C3 000B8      SUBL3   PARTIAL_LENGTH, LENGTH, -(SP)                   ; 3089
                                       08 BC46 9F 000BD      PUSHAB  @SRC_ADDRESS[PARTIAL_LENGTH]                    ; 3088
                                       04 BC46 9F 000C1      PUSHAB  @DEST_UNMAP_ADDR[PARTIAL_LENGTH]                ; 3087
                           0105    C9        03  FB 000C5      CALLS   #3, PAT$WRITE_MEM                               ; 3088
                                 50        01  D0 000CA 7$:  MOVL    #1, R0                                          ; 3090
                                           04  000CD         RET                                                     ; 3093

; Routine Size:  206 bytes,    Routine Base:  _PAT$CODE + 019D
```

```
;  688          3094  1 END
;  689          3095  0 ELUDOM


                                                        .EXTRN  LIB$SIGNAL

;                               PSECT SUMMARY
;
;       Name                    Bytes                           Attributes
;
;  _PAT$CODE                       619  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  .  ABS  .                         0  NOVEC,NOWRT,NORD ,NOEXE,NOSHR,  LCL,  ABS,  CON,NOPIC,ALIGN(0)


;                             Library Statistics
;                                      -------- Symbols --------      Pages       Processing
;       File                           Total    Loaded   Percent      Mapped      Time
;
;  _$255$DUA28:[SYSLIB]LIB.L32;1        18619        13         0        1000        00:01.8



;                                       COMMAND QUALIFIERS
;
;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/VARIANT:1/LIS=LIS$:PATARI/OBJ=OBJ$:PATARI MSRC$:PATARI/UPDATE=(ENH$:PATARI)

; Size:         619 code + 0 data bytes
; Run Time:         00:29.2
; Elapsed Time:     01:33.8
; Lines/CPU Min:     6370
; Lexemes/CPU-Min: 39797
; Memory Used:  201 pages
; Compilation Complete
```

PATARI
LIS

PATCMD
LIS

PATECO
LIS

PATCON
LIS

PATENC
LIS

PATBAS
LIS

PATBLD
LIS